# RWGUI Manual

## GUI for *RWClustering* Application

Author: Akshay Nagendra akshaynag@gatech.edu

Last Revised: April 24th, 2018

## Overview

*RWGUI* is a Python script that generates a GUI for the *RWClustering* application. The python script uses an output of the *RWClustering* application, specifically the *input_graph.dmp* file that holds information for each node and formed cluster. *RWGUI* runs a custom placement algorithm to generate a visual representation of a circuit netlist as a directed acyclic graph (DAG). Once the DAG has been created, a user can view it, as well as the different steps of the Rajaraman-Wong clustering algorithm.

## Quick Run Steps

*RWGUI* is run as part of the csh execution script, *RWCExecute.csh*, if the user supplies a *–gui* flag to the script. If a user wishes to run the GUI by itself, the user can do the following:

1. Navigate to *RWClustering/Python*
2. Open a terminal in this directory and do the following:
    a. If running the GUI on a local UNIX machine, execute the following command: `python RWGUI.py –native`
    b. If running the GUI using an X11 server and SSH: `python RWGUI.py –x11`

**Important Note:** the GUI only supports *RWClustering* runs where the Rajaraman-Wong clustering algorithm is selected, as opposed to the Lawler clustering algorithm

You should see a similar window as the following appear

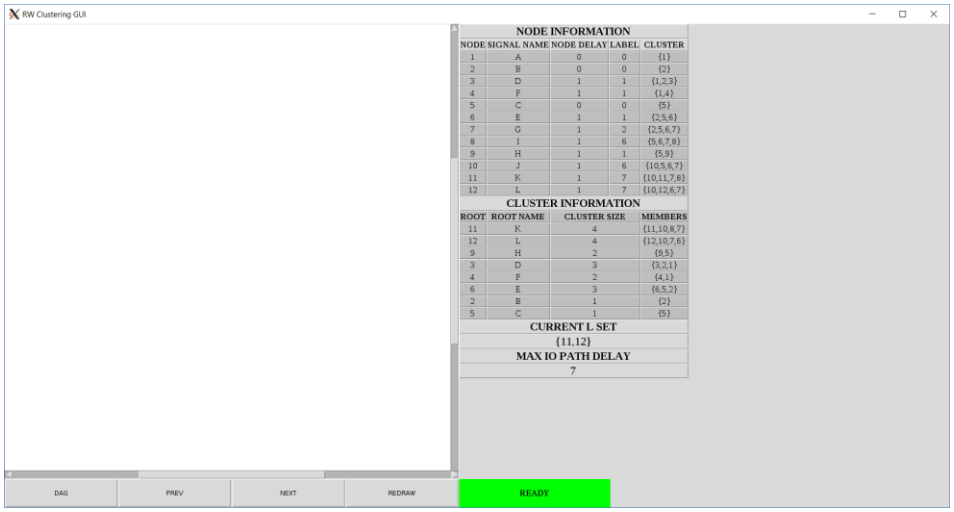


**Figure 1.** Screenshot of *RWGUI* upon successful execution of the *RWGUI.py* python script.

## Using the GUI (General)

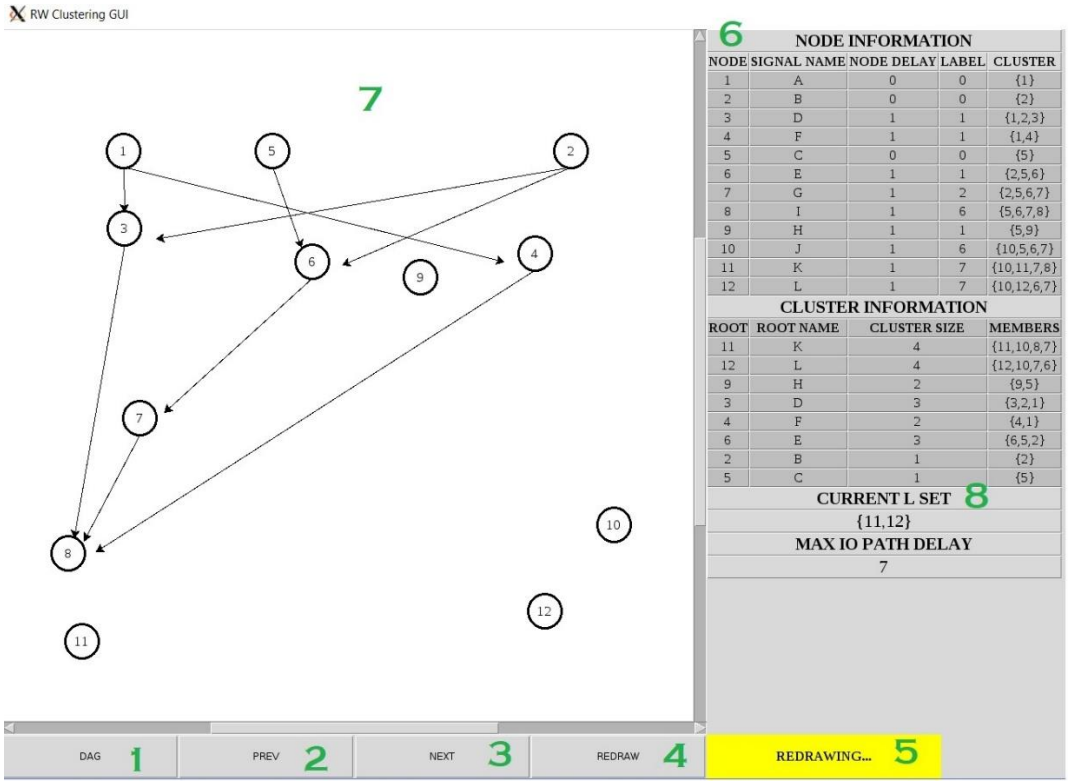


**Figure 2.** Screenshot of *RWGUI* displaying the DAG of the input circuit file and general structure of the GUI.

1. **DAG** Button
    a. Click this button have the DAG for the input circuit display in the canvas (**7**)
2. **PREV** Button
    a. Click this button to either begin the clustering walkthrough or to go to the previous iteration of *L* set traversal to see which cluster was formed at that stage of the clustering algorithm
    b. The current state of the *L* set can be seen in the table at location **8**
3. **NEXT** Button
    a. Click this button either begin the clustering walkthrough or to go to the next iteration of *L* set traversal to see which cluster was formed at that stage of the clustering algorithm
    b. The current state of the *L* set can be seen in the table at location **8**
4. **REDRAW** Button
    a. Click this button to redraw/refresh the screen for whatever is currently being drawn; useful since Python Turtle graphics (used for drawing the graphs) can cause some graphical glitches

5. **STATUS** indicator
    a. This indicator is YELLOW when processing information (e.g. “REDRAWING” or when constructing the DAG or the cluster graph); when the status bar is yellow, please do not click any of the other buttons since Python Turtle graphics does not have great support for handling button callbacks during drawing
    b. When the status bar turns GREEN and says “READY,” feel free to click any button to continue interacting with the GUI
6. Node and Cluster Table
    a. Lists information for each node, including the topological ID, the signal name as it appears in the BLIF file and the *RWClustering* application, the delay, its RW label, and its RW cluster
    b. Lists information for each formed cluster, the signal name for the root of the cluster, and which nodes appear in that cluster
    c. Lists the max IO path delay and the L set at the current stage of the clustering walkthrough
7. Python Turtle Canvas
    a. This is the canvas upon which the DAG and cluster graph is drawn on
    b. Refrain from clicking any button until the status bar (**5**) is green and says “READY”
8. *L* set Indicator
    a. Indicates the current state of the *L* set at this stage of the clustering algorithm

## Walking through the Clustering Algorithm

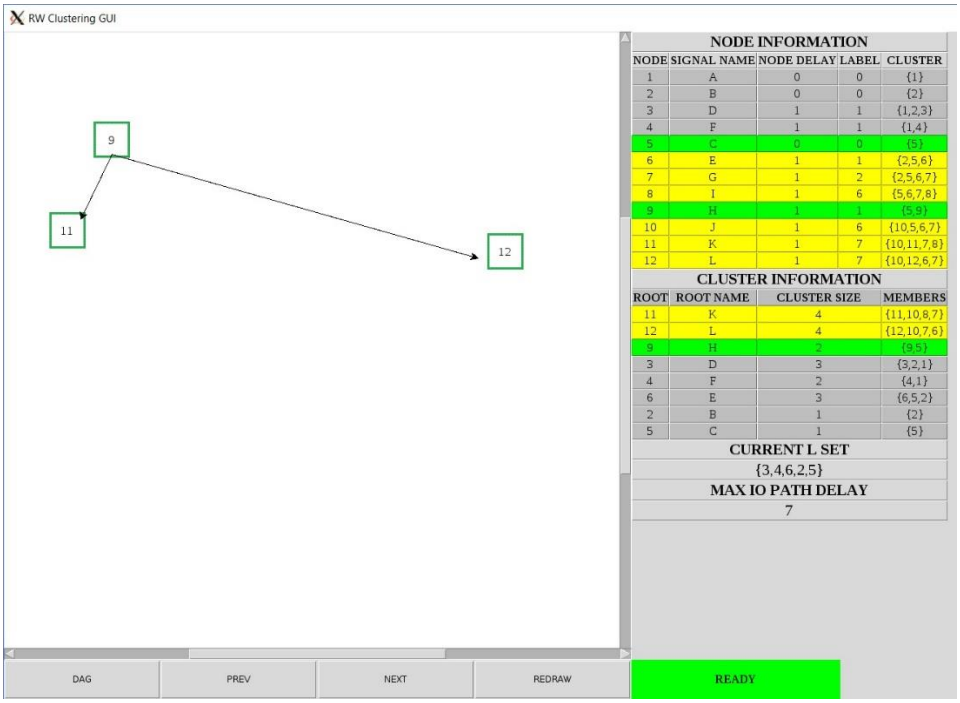


**Figure 3.** Screenshot of *RWGUI* during the clustering algorithm walkthrough.

Clicking on either *PREV* or *NEXT* will begin the clustering algorithm walkthrough. When a cluster is formed, it appears on the canvas as a green square with the topological node ID inside of it as can be seen in Figure 3. At this stage of the clustering algorithm, the *L* set state is displayed in the table on the right which indicates which nodes' clusters are yet to be formed. The current cluster that was just formed is highlighted in green in the **CLUSTER INFORMATION** table and every node that belongs to that cluster is highlighted in green in the **NODE INFORMATION** table. All nodes and clusters that have already been clustered appear in yellow. Once again, an example of the clustering walkthrough in one of the middle stages of the clustering algorithm can be seen in Figure 3. Once the walkthrough is complete (i.e. the user clicks through to the end of the clustering algorithm by clicking *NEXT* repeatedly), the GUI will look like Figure 4.

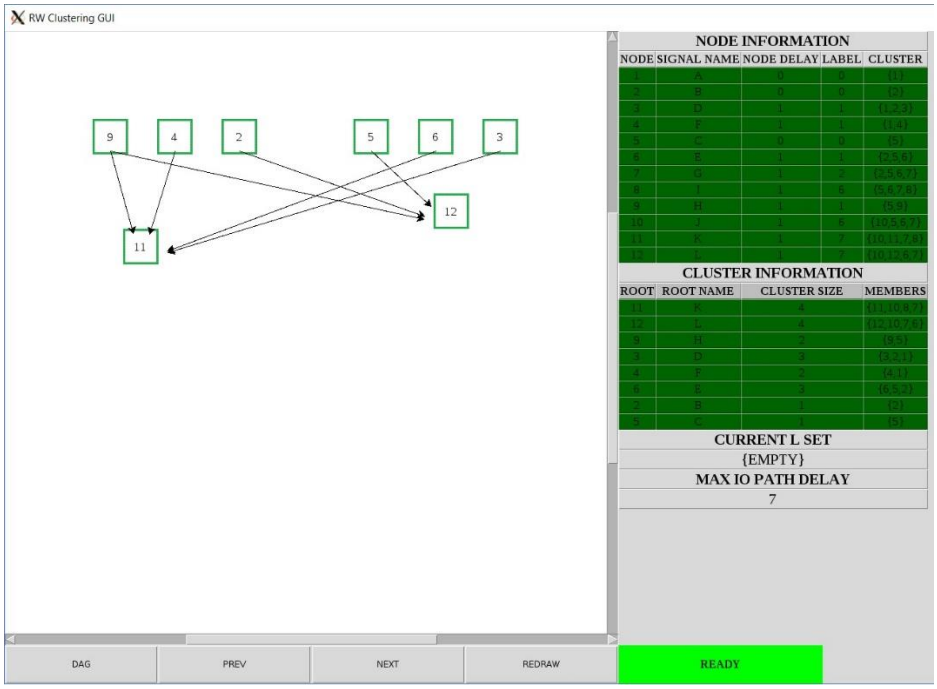


**Figure 4.** Screenshot of *RWGUI* when the walkthrough the clustering algorithm is complete. All node and cluster entries are highlighted in dark green to indicate completion. The *L* set is displayed as *{EMPTY}*, in addition to the clustered DAG being displayed on the canvas.

## Algorithm Overview

### DAG Construction

An initial placement ([x,y] coordinate pair) for each node is generated by traversing through all the nodes in a topological order and placing nodes that appear earlier in the topological order towards the top of the GUI canvas. In order to prevent the edges from crossing through other nodes and thereby affecting the readability of the DAG, a random jitter value is applied to every node's (x,y) coordinate. Once every node has a potential location, a verification function is run to make sure the overlap of edges through nodes is minimized. Once an acceptable placement has

been verified, that placement is finalized and can be viewed by the user, by clicking the “DAG” button. *RWGUI* also reports how many iterations of the custom placement algorithm was required for an acceptable placement and total execution time for the algorithm.

**Cluster Walkthrough**

In Rajaraman-Wong clustering, cluster formation is done by traversing a specific node set, called *L*, which initially holds all the primary outputs in the circuit. As a node from *L* is clustered, all input clusters that have not yet been formed are appended to *L*. The clustering algorithm proceeds until *L* is empty, at which point the clustering algorithm completes. Instead of walking through the full clustering algorithm, the *RWClustering* application prints a trace of the *L* set as the clusters are formed to the *input_graph.dmp* file which is used by *RWGUI* to generate the GUI. In addition to the *L* set trace, the cluster which is formed at each step of the clustering algorithm is also noted, thereby allowing *RWGUI* to simply read the input file and display the correct formed clusters on the screen depending on what step in the clustering algorithm that have stepped to.